(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 11259553 A

(43) Date of publication of application: 24 . 09 . 99

(51) Int. Cl G06F 17/50
G06F 9/06

(21) Application number: 10062555

(22) Date of filing: 13 . 03 . 98

(71) Applicant: OMRON CORP INABATA & CO LTD

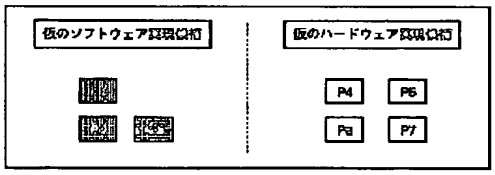
(72) Inventor: KATAOKA SHOICHI
FURUWATARI TOSHIAKI

(54) DESIGN SUPPORTING METHOD FOR SYSTEM WHERE HARDWARE AND SOFTWARE COEXIST

(57) Abstract:

PROBLEM TO BE SOLVED: To automatically decide temporary software part and hardware part from a system specification description and then to automatically optimize the division point of both by evaluation through simulation.

SOLUTION: The system specification description for describing the specifications of a system by the set of the processes of an execution unit is prepared, and based on the system specification description, all the processes are initially divided into temporary hardware realization candidates P4-P7 and temporary software realization candidates P1-P3. At the time, information for indicating priority is attached to the one accompanying the priority in the execution of a processing, and at the time of initial division, the ones to which the information for indicating the priority is attached are turned to the software realization candidates and the others are turned to the hardware realization candidates. Then, the candidates are simulated, execution conditions are stored as profiling information, a change candidate from the respective candidates is selected based on it, the improvement rate of a chip area and an effective speed is evaluated further and thus, the division point is automatically corrected to the optimum one.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－259553

(43)公開日　平成11年(1999)９月24日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　17/50		Ｇ０６Ｆ　15/60	６５４Ｍ
9/06	５３０	9/06	５３０Ｕ

審査請求　未請求　請求項の数７　ＯＬ　（全 11 頁）

(21)出願番号　特願平10－62555

(22)出願日　平成10年(1998)３月13日

(71)出願人　000002945
オムロン株式会社
京都府京都市右京区花園土堂町10番地

(71)出願人　595144466
稲畑産業株式会社
大阪府大阪市中央区南船場１丁目15番14号

(72)発明者　片岡　庄一
京都府京都市下京区木津屋橋通西洞院東入る東塩小路町606番地　オムロンソフトウェア株式会社内

(72)発明者　古渡　俊明
東京都台東区上野３丁目３番８号　株式会社アイ・ケイ・テクノロジー内

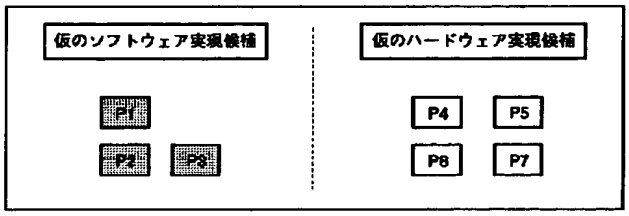
(74)代理人　弁理士　小森　久夫

(54)【発明の名称】　ハードウエアとソフトウエアの混在するシステムの設計支援方法

(57)【要約】

【課題】システム仕様記述から自動的に仮のソフトウエア部分をハードウエア部分を決め、その後シミュレートによる評価により、双方の分割点を自動的に最適化していく。

【解決手段】システムの仕様を実行単位のプロセスの集まりで記述したシステム仕様記述を作成し、このシステム仕様記述に基づいて、全プロセスを仮のハードウエア実現候補Ｐ４～Ｐ７と、仮のソフトウエア実現候補Ｐ１～Ｐ３とに初期分割する。このとき、処理の実行に優先度の伴うものについては優先度を示す情報が付されており、初期分割に際しては、この優先度を示す情報の付されたものをソフトウエア実現候補とし、それ以外をハードウエア実現候補とする。次に、これらの候補をシミュレーションして実行状況をプロファイリング情報として記憶し、これに基づいてそれぞれの候補からの変更候補を選択して、さらに、チップ面積や実効速度の改善率を評価することにより、分割点を自動的に最適なものに修正する。



プロセスの初期分割

【特許請求の範囲】

【請求項１】　システムの仕様を実行単位のプロセスの集まりで記述したシステム仕様記述を作成し、このシステム仕様記述に基づいて、全プロセスを仮のハードウエア実現候補と仮のソフトウエア実現候補に初期分割し、次いで、これらの候補をシミュレーションして実行状況をプロファイリング情報として収集し、このプロファイリング情報に基づいて、仮のハードウエア実現候補からソフトウエア実装への変更候補、または、仮のソフトウエア実現候補からハードウエア実装への変更候補を選択し、これらの変更候補によるチップ面積と実行速度の改善率を評価して、上記各候補の分割点を修正することを特徴とする、ハードウエアとソフトウエアの混在するシステムの設計支援方法。

【請求項２】　プロセスは、実行のための優先度の情報を含み、初期分割は、この優先度の含むプロセスをソフトウエア実現候補、含まないプロセスをハードウエア実現候補として自動的に行うようにした、請求項１記載の、ハードウエアとソフトウエアの混在するシステムの設計支援方法。

【請求項３】　前記変更候補の最小単位は、プロセス内の実行ブロックである、請求項１または２記載の、ハードウエアとソフトウエアの混在するシステムの設計支援方法。

【請求項４】　プロファイリング情報は、ソフトウエア実現候補の各プロセス毎の実行ブロックに属する変数へのアクセス回数を含み、アクセス回数が一定以上の変数が属する実行ブロックをハードウエア実装への変更候補とする、請求項１～３のいずれかに記載の、ハードウエアとソフトウエアの混在するシステムの設計支援方法。

【請求項５】　プロファイリング情報は、ハードウエア実現候補の各プロセス毎の実行ブロックに属する変数へのアクセス回数を含み、アクセス回数が一定以下の変数が属する実行ブロックをソフトウエア実装への変更候補とする、請求項１～３のいずれかに記載の、ハードウエアとソフトウエアの混在するシステムの設計支援方法。

【請求項６】　各候補の分割点を修正するとき、速度改善値のより高いハードウエア実装変更候補と、その候補と実装面積の均衡をとり得るソフトウエア実装変更候補の組とを選び、これらの候補をそれぞれハードウエア実装分、ソフトウエア実装分に変更することで分割点修正を行うようにした、請求項１～５のいずれかに記載の、ハードウエアとソフトウエアの混在するシステムの設計支援方法。

【請求項７】　各候補の分割点を修正するとき、面積改善値のより高いソフトウエア実装変更候補と、その候補と実行時間の均衡をとり得るハードウエア実装変更候補の組とを選び、これらの候補をそれぞれソフトウエア実装分、ハードウエア実装分に変更することで分割点修正を行うようにした、請求項１～５のいずれかに記載の、ハードウエアとソフトウエアの混在するシステムの設計支援方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ハードウエアとソフトウエアの混在するシステムの設計支援方法に関し、特に、システム仕様からハードウエアとソフトウエアの分割を自動化、且つ最適化出来る支援方法に関する。

【０００２】

【従来の技術】今日、ＡＳＩＣの高集積化は数千万トランジスタの１チップ化を可能にしており、ＣＰＵ、ファームウエア、ドライバなどに加えて、通信回路、入出力バス・インターフェイス回路などの周辺回路なども全て一つのＡＳＩＣ上に納まることを可能にしている。

【０００３】このようなＡＳＩＣは、いわゆるシステムＡＳＩＣと称される。

【０００４】ところでシステムＡＳＩＣには、プロセッサとペリフェラルとメモリが実装され、プロセッサとメモリを利用して従来のソフトウエアが実装される形になる。ここで一つの機能をハードウエアで実現するかソフトウエアで実現するかにおいて、速度とチップ面積のトレードオフが内在している。つまり、ハードウエアはソフトウエアに対して相対的に速度を上げるがチップ面積も押し上げる。また、ソフトウエアはその反対の特性を持つ。

【０００５】設計に際して、重要なことは、そのトレードオフの関係を考慮し、与えられたチップ面積上で最大の速度を上げ得る双方の混在化、すなわち、ハードウエアとソフトウエアの分割点の最適化を図ることである。

【０００６】従来、この要求に対しては、熟練した人間の経験及び勘を頼りに、ある機能を実現するために、まず、ハードウエア部分とソフトウエア部分を決め、それぞれが独立して設計された後に協調動作シミュレーションなどを行って、不都合な部分を抽出し、さらに、再設計、シミュレーションを繰り返していく手法がとられている。

【０００７】また、特開平９－１６０９４９号では、まず、ハードウエア部分とソフトウエア部分を決めた後に、ソフトウエア部分のハードウエア化可能部分を探索し、可能な部分があれば、ソフトウエアを行うという手法が提案されている。

【０００８】

【発明が解決しようとする課題】しかしながら、上記いずれの手法も、最初に、ハードウエア部分とソフトウエア部分を人間が決めているため最終結果は必ずしも、速度とチップ面積の観点で最適なものとはならなかったり、また、ハードウエア部分とソフトウエア部分そのものを分割すること自体、極めて困難であった。また、通常、ハードウエア部分はＨＤＬ言語（ハードウエア回路用言語）で書かれてこれをロジックで表現する一方、ソ

フトウエア部分はＣ言語などで書かれるため、従来の手法では両者は別扱いする必要があり、これらを協調的に同時並行開発することが基本的に無理であった。すなわち、双方の部分の交換を同時並行的に行ったり、その交換直後に結果をシミュレートする等、協調と同時並行開発を行うことは出来なかった。本発明の目的は、システム仕様記述から自動的に仮のソフトウエア部分をハードウエア部分を決め、その後シミュレートによる評価により、双方の分割点を自動的に最適化していくことの出来る、ハードウエアとソフトウエアの混在するシステムの設計支援方法を提供することにある。

【０００９】

【課題を解決するための手段】本出願の請求項１に係る発明は、システムの仕様を実行単位のプロセスの集まりで記述したシステム仕様記述を作成し、このシステム仕様記述に基づいて、全プロセスを仮のハードウエア実現候補と仮のソフトウエア実現候補に初期分割し、次いで、これらの候補をシミュレーションして実行状況をプロファイリング情報として収集し、このプロファイリング情報に基づいて、仮のハードウエア実現候補からソフトウエア実装への変更候補、または、仮のソフトウエア実現候補からハードウエア実装への変更候補を選択し、これらの変更候補によるチップ面積と実行速度の改善率を評価して、上記各候補の分割点を修正することを特徴とする。

【００１０】本発明では、システム仕様を、実行単位のプロセスの集合として記述することを出発点とする。この実行単位のプロセスとは、ＣＰＵで実行される並列処理単位で、且つハードウエアでもソフトウエアでも実現可能なものを言う。つまり、システム仕様記述上は、各プロセスは見かけ上、ハードウエアでもソフトウエアでも実現可能なようになっている。

【００１１】次に、システム仕様記述から、ハードウエア部分とソフトウエア部分の実現候補を自動的に決めて分割する。プロセスには種々の理由からハードウエアで実現する方が望ましいものとソフトウエアで実現する方が望ましいものが存在するから、これを自動判断し、且つそれに基づきハードウエア部分とソフトウエア部分とを仮に決め、残りの部分については、ランダムに決めることが出来る。

【００１２】次に、これらの仮の候補に対してシミュレーションを行い、実行状況をプロファイリング情報として収集する。シミュレーションに際しては、ハードウエア部分及びソフトウエア部分の各プロセスの実行時間、より詳細にはプロセスに含まれる実行ブロックの実行時間をクロックで計測することが出来る。これにより、実行状況としてハードウエア部分及びソフトウエア部分の各プロセスの実行速度を計測し、これをプロファイリング情報として収集することが出来る。

【００１３】上記プロファイリング情報によれば、各プロセスの実行速度がわかるから、実効速度の遅いソフトウエア部分のプロセスに対してはハードウエアに変更することが考えられる。この場合、ハードウエアへの変更により速度は上がるがチップ面積が増加することが実際に変更するか否かの評価対象となる。また、実行速度の遅いハードウエア部分のプロセスに対してはソフトウエアに変更することが考えられる。この場合、ソフトウエアへの変更によりチップ面積は減少するが速度が低下することが実際に変更するか否かの評価対象となる。

【００１４】このようにして、変更候補によるチップ面積と実行速度の改善率を評価して、各候補の分割点を修正する。

【００１５】なお、分割点の修正後、再びその分割によるハードウエア部分とソフトウエア部分でシミュレーションを行い、上記と同様な評価を繰り返すことも出来る。

【００１６】本出願の請求項２に係る発明は、プロセスは、実行のための優先度の情報を含み、初期分割は、この優先度の含むプロセスをソフトウエア実現候補、含まないプロセスをハードウエア実現候補として自動的に行うようにしたものである。

【００１７】プロセスは相互に通信を行いながら並行動作することが基本であるため（リアルタイム処理）、ソフトウエア部分のプロセスについては、ＣＰＵの実行権を得るための指標が予めプロセス中に含まれているのが望ましい。この請求項２に係る発明では、その指標として処理の優先度の情報を与えるものである。したがって、初期分割を行うときに、優先度の情報があるプロセスをソフトウエア実現候補として、それ以外をハードウエア実現候補として分割する。

【００１８】本出願の請求項３に係る発明は、前記変更候補の最小単位は、プロセス内の実行ブロックであることを特徴とする。

【００１９】１つのプロセス内には１つ以上の実行ブロックが含まれているが、その中には処理速度の高速性が要求されるものと要求されないものが混在することがある。したがって、プロセス単位を入れ替えとするよりも実行ブロック単位を入れ替え対象とする方が最適化を図る上で好ましいと言える。この場合、実行ブロック単位が最小単位であり、これ以上のかたまりを単位として扱うことも可能である。本出願の請求項４に係る発明は、プロファイリング情報は、ソフトウエア実現候補の各プロセス毎の実行ブロックに属する変数へのアクセス回数を含み、アクセス回数が一定以上の変数を含むプロセスをハードウエア実装への変更候補とするものである。また、請求項５に係る発明は、プロファイリング情報は、ハードウエア実現候補の各プロセス毎の実行ブロックに属する変数へのアクセス回数を含み、アクセス回数が一定以下の変数が属する実行ブロックをソフトウエア実装への変更候補とするものである。

【００２０】システム仕様を表すプロセスはそれ自体シミュレータで実行可能な形式でないから、これを翻訳してシミュレーションコード（プログラム）を起こしてシミュレータにかけるが、このとき、シミュレータにおいて、各プロセスに含まれる実行ブロックに属する変数へのアクセス回数をプロファイリング情報として集める。そして、ソフトウエア実現候補のうちアクセス回数の多いブロックはハードウエア実装への変更候補とし、ハードウエア実現候補のうちアクセス回数の少ないブロックはソフトウエア実装への変更候補とする。すなわち、ソフトウエア部分で頻繁にアクセスするものはシステムのスループットを悪くすると推定でき、一方、ハードウエア部分であまりアクセスしないものはチップ面積の無駄遣いと推定されるから、このような部分については変更の候補とする。

【００２１】本出願の請求項６に係る発明は、各候補の分割点を修正するとき、速度改善値のより高いハードウエア実装変更候補と、その候補と実装面積の均衡をとり得るソフトウエア実装変更候補の組とを選び、これらの候補をそれぞれハードウエア実装分、ソフトウエア実装分に変更することで分割点修正を行うようにしたものである。　このように実装面積の均衡がとれるように、速度改善値のより高いハードウエア実装変更候補とソフトウエア実装変更候補を選択することで、面積を一定に保ちながら実行速度の点で有利な分割点を自動的に得ることが出来る。

【００２２】本出願の請求項７に係る発明は、各候補の分割点を修正するとき、面積改善値のより高いソフトウエア実装変更候補と、その候補と実行時間の均衡をとり得るハードウエア実装変更候補の組とを選び、これらの候補をそれぞれソフトウエア実装分、ハードウエア実装分に変更することで分割点修正を行うようにしたものである。

【００２３】このように実行時間の均衡がとれるように、面積改善値のより高いソフトウエア実装変更候補とハードウエア実装変更候補を選択することで、実行時間を一定に保ちながら実装面積の点で有利な分割点を自動的に得ることが出来る。

【００２４】

【発明の実施の形態】図１は、本発明の実施形態である、ハードウエアとソフトウエアの混在するシステムの設計支援装置の構成図である。

【００２５】この支援装置は、システム仕様記述１で書かれたプロセスに基づいて協調合成システム２において、ハードウエア部分とソフトウエア部分とに分割し、ハードウエア部分については動作合成システム３、論理合成システム４でＨＤＬ言語に変換しつつハードウエアロジック回路を自動作成する。また、ソフトウエア合成システム５は、ソフトウエア部分からＣ言語などによるプログラムコードを自動作成する。スタティックタイミング検証システム６は、このプログラムコードやハードウエアロジック回路の静的動作（スタティック動作）の検証を行い。合成結果表示システムは、上記一連の手順に伴う結果を適宜表示する。

【００２６】システム仕様記述のプロセスの１例を図２に示す。

【００２７】ここでは、プロセス名を「sample」としている。

【００２８】第２行は、整数（int ）入力データ端子として、iData を定義する。第３行は、整数出力端子として、oDATA を定義し且つ初期値が０であることを示す。第４行、第５行は、制御入力端子Strtと制御出力端子ackStrt を定義し、制御出力端子ackStrt の初期値がFALSE であることを示す。第６行以下はプロセス実行部分である。要約すれば、制御入力端子StrtがTRUEのときに、制御出力端子ackStrtをTRUEにセットし、変数counter のインクリメント値が入力データ端子iData の値よりも大きければ変数counter をリセットし、大きくなければ出力データ端子ODataの値を、iData 値にcounter 値を加えた値とする。

【００２９】このようなプロセスは、基本的にソフトウエアでもハードウエアでも実現が可能である。

【００３０】図１の協調合成システム２は、上記プロセスを読み込んで、全プロセスをハードウエア部分（ハードウエア実現候補）とソフトウエア部分（ソフトウエア実現候補）とに初期分割し、これをシミュレーションして相互の各プロセス部分のソフトウエア化またはハードウエア化が適正か否かを評価し、評価結果にしたがって、１部の入れ替えを行い、さらに、必要に応じてその入れ替えた結果で再度シミュレーションを行う動作を繰り返す。

【００３１】システム仕様記述１に書かれる各プロセスは、本実施形態では初期分割しやすいように、ソフトウエア部分の候補となるプロセスに処理の優先度を表す情報がつけ加えられる。図３にその状態を示す。なお、ソフトウエアのプロセス群は１つのＣＰＵで処理が実行される限り、各プロセス間で通信をするときに処理の優先度が必要となることがある。優先度を表す情報はこのためのものである。ハードウエアのプロセス群はＣＰＵにより処理されるものではないから、通常は処理に優先度を必要としない。もちろん、この情報が付加されていてもこれをハードウエア部分で構成することは可能である。

【００３２】図３において、PROCA 0 、PROCB 1 は、前者のプロセス優先度が「０」、後者のそれが「１」であることを示している。優先順位は前者の方が一つ高い。

【００３３】協調合成システム２は、上記の優先度を表す情報が付加されているプロセスを仮のソフトウエア実現候補とする。また、その他のプロセスを仮のハードウエア実現候補とする。初期分割はこのようにして行われ

る。図４に初期分割した状態を示している。Ｐ１～Ｐ３は優先度を表す情報を持つプロセスであるため、ソフトウエア実現候補とされる。Ｐ４～Ｐ７は優先度を表す情報を持たないプロセスであるため、ハードウエア実現候補とされる。

【００３４】なお、この段階で、図４のように分類された仮のソフトウエア実現候補と仮のハードウエア実現候補に対して、面積（ソフトウエア部分についてはプログラムステップ数）と、実行時間が初期見積もりデータとして保存される。これらの値は、図１のソフトウエア合成システム５と動作合成システム３により求められる。すなわち、プログラムステップ数及び面積は、ソフトウエア部分のプログラムステップ数及びハードウエア部分のＨＤＬ言語から分析した回路により、また、実行時間は、予め設定されているクロック時間と上記プログラムステップ数及び回路の遅延時間とにより求められる。

【００３５】次に、上記の仮のハードウエア実現候補と仮のソフトウエア実現候補をシミュレータにかけて、実行状況のプロファイリング情報を収集する。

【００３６】図５は、プロセスの一般構造を示している。システム仕様記述の中のプロセスの定義には、図５に示すように、さらに小さな機能の定義や、場合分け等のプログラム記述がある。これらの定義やプログラム記述の中で、分岐やブロック化によってまとめられる実行の単位を、ここでは実行ブロックと呼ぶ。一つのプロセス中、処理の高速化を要求される実行ブロックがあれば、その反対に要求されない実行ブロックもある。シミュレータは、各実行ブロックについて、処理の高速化が要求されるかどうかを判断するためのプロファイリング情報を作成するためのものである。プロファイリング情報は、各実行ブロックに含まれる変数へのアクセス回数を含んでいる。すなわち、アクセス回数が一定以上であれば、その変数が属する実行ブロックについては処理の高速化が必要であると推定する。反対に、アクセス回数が一定以下であれば、その変数が属する実行ブロックについては処理の高速化は必要でないと推定する。このように、各ブロックに属する変数へのアクセス回数をプロファイリング情報として収集することにより、各ブロックに対し、処理の高速化が必要か否かを知ることが出来るようになる。そして、仮のソフトウエア実現候補内の実行ブロックに処理の高速化が必要なものがあると判断すれば、その実行ブロックをハードウエア実装への変更候補に設定する。また、仮のハードウエア実現候補内の実行ブロックに処理の高速化が必要でないものがあると判断すれば、その実行プロセスをソフトウエア実装への変更候補に設定する。

【００３７】図１の協調合成システム２に含まれるシミュレータは、システム仕様記述からシミュレーションコードを作成してシミュレーションを実行出来るようにし、また、そのときに各変数がどの実行ブロックに属するかを示すスコープ情報を作成し、これを各実行ブロックの従属関係を示す木（ツリー）構造で管理する。木構造の各実行ブロックには変数のアクセス回数をカウントするカウンタが埋め込まれ、シミュレーションコードの実行に従って、このカウンタを加算していく。これにより、実行ブロック毎の変数のアクセス回数をカウントすることが出来るから、この結果をプロファイリング情報として出力する。図６にシミュレーションを行うときの状態を示す。

【００３８】次に、上記のようにして得られたプロファイリング情報から、ソフトウエア実現候補の中のプロセスのうち所定の実行ブロックをハードウエア実装への変更候補に選び、また、ハードウエア実現候補の中のプロセスのうち所定の実行ブロックをソフトウエア実装への変更候補に選ぶ。以下、この選択手法について説明する。

【００３９】（１）ソフトウエアからハードウエアへの入れ替え候補の選択

ソフトウエア実現候補のプロセスに含まれる実行ブロックのうち、シミュレーション実行後のアクセス回数の多い実行ブロックのランク付けを行い、高いランク（高アクセス回数）を持つ実行ブロックのそれぞれについて、ハードウエアとして切り出した場合のオーバーヘッドを加算したハードウエア増加面積を算出する。オーバーヘッドとは、ハードウエアとして切り出した場合に、余分に必要となるハードウエア部分（例えば、アドレス設定レジスタなど）をいう。この演算は、最高アクセス回数を持つ実行ブロックから最高アクセス回数の２分の１の回数を持つ実行ブロックまでを対象とする。

【００４０】（２）ハードウエアからソフトウエアへの入れ替え候補の選択

ハードウエア実現候補のプロセスに含まれる実行ブロックのうち、シミュレーション実行後のアクセス回数の少ない実行ブロックのランク付けを行い、高いランク（低アクセス回数）を持つ実行ブロックのそれぞれについて、ソフトウエアとして切り出した場合のオーバーヘッドを減算したハードウエア減少面積を算出する。この演算は、最低アクセス回数を持つ実行ブロックから最高アクセス回数の２分の１の回数を持つ実行ブロックまでを対象とする。

【００４１】上記（１）（２）で選択した入れ替え候補について、以下のアルゴリズムにより入れ替えを確定する。

【００４２】（Ａ）上記（２）の入れ替え候補の実行ブロック（以下、ソフトウエア実装変更候補モジュールと呼ぶ）を、ソフトウエア合成システム５（図１参照）を用いて処理し、ソフトウエア見積もりデータを取得する。ソフトウエア見積もりデータには、プログラムステップ数と実行時間が含まれる。

【００４３】（Ｂ）上記（１）の入れ替え候補の実行ブ

ロック（以下、ハードウエア実装変更候補モジュールと呼ぶ）を、動作合成システム３（図１参照）を用いて処理し、ハードウエア見積もりデータを取得する。ハードウエア見積もりデータには、実装の面積と実行時間が含まれる。

【００４４】（Ｃ）保持していた初期分割におけるソフトウエア見積もりデータ（初期見積もりデータ）とソフトウエア実装変更候補モジュールのソフトウエア見積もりデータとを比較する。比較により、面積改善率と速度改悪率を求める。前者の面積改善率は、ソフトウエア実装変更候補モジュールの実現により減少する面積の改善率であり、速度改悪率は、ソフトウエア実装変更候補モジュールの実現により遅くなる速度の改悪率である。その結果、面積改善率／速度改悪率が１．０以下のモジュールを非効率実装モジュールとし、これをソフトウエア実装変更候補モジュールから除去する。

【００４５】（Ｄ）保持していた初期分割におけるハードウエア見積もりデータ（初期見積もりデータ）とハードウエア実装変更候補モジュールのハードウエア見積もりデータとを比較する。比較により、速度改善率と面積改悪率を求める。前者の速度改善率は、ハードウエア実装変更候補モジュールの実現により速くなる速度の改善率であり、面積改悪率は、ハードウエア実装変更候補モジュールの実現により増大する面積の改悪率である。その結果、速度改善率／面積改悪率が１．０以下のモジュールを非効率実装モジュールとし、これをハードウエア実装変更候補モジュールから除去する。

【００４６】（Ｅ）ソフトウエア実装変更候補モジュールにおいて、面積改善率の大きなものから実装変更優先度を付与する。

【００４７】（Ｆ）ハードウエア実装変更候補モジュールにおいて、速度改善率の大きなものから実装変更優先度を付与する。

【００４８】（Ｇ）ハードウエア実装変更候補モジュールの実装変更優先度の高いものから順に参照し、その面積増加分を補うソフトウエア実装変更候補モジュールの組み合わせを探索する。つまり、面積が増えないように全体として処理速度が改善される組み合わせを探索する。この探索により、可能な限り実装変更優先度の高いものを組み合わせ、同一実装面積でさらに処理速度が改善される組み合わせを探る。

【００４９】上記（Ｇ）の処理は、一定の面積内で速度が最小となる方向を目指す。この具体的な内容について図７～図９を参照して説明する。

【００５０】図７は、（Ｅ）の処理を終えたときのソフトウエア実装変更候補モジュールのソート結果を示している。また、図８は、（Ｆ）の処理を終えたときのハードウエア実装変更候補モジュールのソート結果を示している。図７において、１列目の「Ｈ１」・・・はソフトウエア実装変更候補モジュールの識別番号、２列目の「面積」と「実行時間」は、初期見積もりデータとして得られた当初のハードウエア部分での面積とその部分での実行時間、３列目の「ステップ数」と「実行時間」は、ソフトウエア実装変更候補モジュールについて（Ａ）の処理により得られたステップ数と実行時間、４列目の「減少面積」はソフトウエアからハードウエアへ変換した場合の減少面積、すなわち、２列目の面積から３列目のステップ数面積換算値を引いた値、５列目の「既変」は候補から実際に変換されたことを示すフラグ、６列目の「今回」は変更候補となったことを示すフラグである。また、図８において、１列目の「Ｓ１」・・・はハードウエア実装変更候補モジュールの識別番号、２列目の「ステップ数」と「実行時間」は、初期見積もりデータとして得られた当初のソフトウエア部分でのステップ数と実行時間、３列目の「面積」と「実行時間」は、ハードウエア実装変更候補モジュールについて（Ｂ）の処理により得られた面積と実行時間、４列目の「増加面積」はハードウエアへと変換した場合の増加面積すなわち、３列目の面積から２列目のステップ数面積換算値を引いた値、５列目の「既変」は候補から実際に変換されたことを示すフラグ、６列目の「今回」は変更候補となったことを示すフラグである。

【００５１】図９は、図７及び図８のソート結果を使って上記（Ｇ）の処理を実際に行うフローチャートである。

【００５２】変数（カウンタ）ｎは図８に示すハードウエア実装変更候補モジュールのポインタを示し、変数（カウンタ）ｍは図９に示すソフトウエア実装変更候補モジュールのポインタを示す。

【００５３】ＳＴ１では、ハードウエア実装変更候補モジュールの中で最も速度改善値の高いものを指定する。ＳＴ２において、そのモジュールの増加面積を読む。ＳＴ３では、ソフトウエア実装変更候補モジュールの中で最も面積改善値の高いものを指定し、ＳＴ５でそのモジュールの減少面積を読み込む。

【００５４】変数「比較ゲート数」は、ソフトウエア実装変更候補モジュールの減少面積の足し込んだ値を示す。ＳＴ５において読み込んだソフトウエア実装変更候補モジュールの減少面積がＳＴ７において比較ゲート数に加算されていく。その加算値がＳＴ２で読み込んだハードウエア実装変更候補モジュールの増加面積に等しくなるまで、ＳＴ９を経由してソフトウエア実装変更候補モジュールの減少面積が比較ゲート数に加算されていく。ＳＴ８においてこれらの値が等しくなると、ＳＴ１０において、該当のソフトウエア実装変更候補モジュールの「既変」フラグをオンにし、実際のソフトウエア実装分として確定する。また、ＳＴ１１において、ｎで指定しているハードウエア実装変更候補モジュールの「既変」フラグをオンにして、実際のハードウエア実装分として確定する。なお、実装面積の均衡をとる場合に、ハ

ードウエア実装変更候補モジュールの実装採用による速度改善は、速度改善率の高いものから実装採用となるため、その組み合わせとなるソフトウエア実装変更候補モジュールの実装採用による速度改悪を十分にカバーする。

【００５５】以上のｎ２～ｎ１１までの動作をｎが所定値になるまで連続して処理することにより、チップ面積を増加させずに可能な限り処理速度の向上を図ることが出来る。

【００５６】なお、ＳＴ２において１つのハードウエア実装変更候補モジュールの増加面積を読み出すようにしているが、このステップで上位から複数のハードウエア実装変更候補モジュールの面積加算値を読み出し、これに対応する面積減少値を持つソフトウエア実装変更候補モジュールを選ぶようにすることも出来る。また、反対に、最初にソフトウエア実装変更候補モジュールの面積減少値を読み出し、これに対応する面積増加値を持つハードウエア実装変更候補モジュールを選ぶようすることも出来る。

【００５７】また、上記の処理を一通り行った後、再度、最初から同じ動作を繰り返すことも可能である。何度も繰り返すことにより、さらに、速度向上を図ることが出来るようになる。さらに、チップ面積を増やすことが可能な場合には、その面積に対応する増加面積を有するハードウエア実装変更候補モジュールをハードウエア実装分にすることが出来る。

【００５８】以上の処理により、仮のソフトウエア実装変更候補と仮のハードウエア実現候補とに初期分割したプロセスにおいて、チップ面積を大きくすることなく、速度改善出来るように自動的に各プロセス内の実行ブロックに対して、ソフトウエア実装への変更またはソフトウエア実装への変更を行うことが出来る。

【００５９】図１０～図１２は、本発明の他の実施形態の動作を示している。図７～図９は、一定の面積内で実行速度が最小となる方向に各モジュールを変更する場合の操作内容を示しているが、一定の速度内で面積が最小となる方向に各モジュールを変更することも可能である。図１０～図１２はこの場合の動作を示す。

【００６０】すなわち、図１０に示すように、面積改善値でソートした各ソフトウエア実装変更候補モジュールに、ハードウエアからソフトウエアへ変換するときの増加時間、すなわち、２列目の実行時間から３列目の実行時間を引いた値を付加し、図１１に示すように、速度改善値でソートした各ハードウエア実装変更候補モジュールに、ソフトウエアからハードウエアへ変換するときの減少時間、すなわち、３列目の実行時間から２列目の実行時間を引いた値を付加した状態で、図１２のフローチャートを実行する。

【００６１】図１２では、変数（カウンタ）ｎで図１０のソフトウエア実装変更候補モジュールを先頭から順にポインティングし、変数（カウンタ）ｍで図１１のハードウエア実装変更候補モジュールを先頭から順にポインティングする。ＳＴ２１でｎで示されるソフトウエア実装変更候補モジュールの増加時間に均衡がとれる減少時間を持つハードウエア実装変更候補モジュールを図１１から探し、それらをそれぞれソフトウエア実装分とハードウエア実装分とする。あとの処理は図９と同様である。

【００６２】このように、上の２つの実施形態では、それぞれ、実装面積や実行時間の均衡をとりながらモジュールを交換して、一定の面積で最も速度が最小となる分割点や一定の速度で最も面積が最小となる分割点を自動的に求めることが出来る。

【００６３】

【発明の効果】本発明によれば、最初に、システム仕様記述である実行単位のプロセスの全部を仮のハードウエア実現候補と仮のソフトウエア実現候補に自動分割しているため、分割に人間の経験や勘を必要せず、熟練技術者でなくとも分割が可能である。また、仮の分割点から、それぞれへの変更候補を選択して、チップ面積と実行速度の改善率を評価し、上記分割点の修正を行うようにしているため、システムの最適分割点を自動的に短時間で求めることが出来る。このため、システムの同時並行的開発が可能となる。

【００６４】また、上記仮の候補への自動分割を優先度情報の有無により行うことにより、大きく誤った仮分割となることがなく、分割そのものも非常に簡単となり短時間で出来る。

【００６５】また、１つのプロセスには複数の実行ブロックがあり、各ブロックによりソフトウエアが適切なものやハードウエアが適切なものがあるため、変更候補の最小単位をプロセス全体ではなくプロセス内の実行ブロックとすることで、より適切な分割が可能になる。

【００６６】また、変数へのアクセスが多いときはハードウエア化がより適切なものと考えられ、反対に変数へのアクセスが少ないときはソフトウエア化が適切と考えられるから、プロファイリング情報として、各プロセスに含まれる実行ブロック内の変数へのアクセス回数を含ませれば、変更候補の選択がより正しい方向のものとなる。また、速度改善値の高いハードウエア実装変更候補とそれに実装面積の均衡をとり得るソフトウエア実装変更候補の組を選び、これらをそれぞれハードウエア実装分とソフトウエア実装分に変換する分割点修正を行うことで、一定の面積で速度が最小となる分割点を自動的に見いだすことが出来る。

【００６７】また、面積改善値の高いソフトウエア実装変更候補とそれに実行時間の均衡をとり得るハードウエア実装変更候補の組を選び、これらをそれぞれソフトウエア実装分とハードウエア実装分に変換する分割点修正を行うことで、一定の時間で面積が最小となる分割点を

13

自動的に見いだすことが出来る。

【図面の簡単な説明】

【図１】本発明の実施形態である、ハードウエアとソフトウエアの混在するシステムの設計支援装置の構成図。

【図２】プロセスの集まりで書かれるシステム仕様記述の一例を示す図。

【図３】システム仕様記述におけるプロセスの優先度指定を示す図。

【図４】プロセスの初期分割例を示す図。

【図５】プロセス内の実行ブロックを示す図。

【図６】シミュレータによるシミュレーション動作を説明する図。

14

＊【図７】ソフトウエア実装変更候補モジュールのソート結果を示す図。

【図８】ハードウエア実装変更候補モジュールのソート結果を示す図。

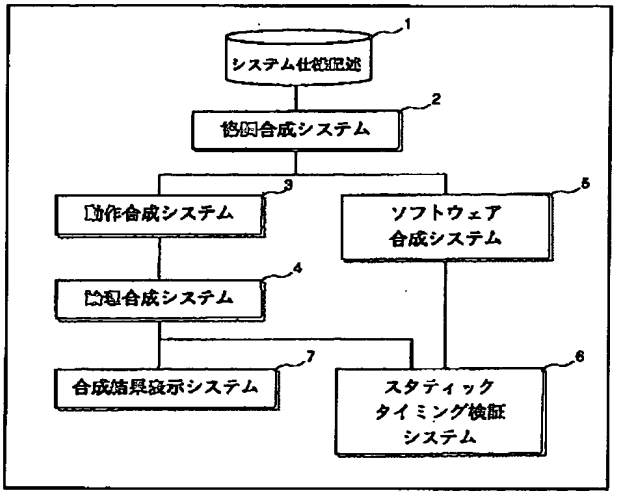
【図９】速度が最小となる方向でモジュール変換を行う手順を示すフローチャート。

【図１０】ソフトウエア実装変更候補モジュールのソート結果を示す図。

【図１１】ハードウエア実装変更候補モジュールのソート結果を示す図。

【図１２】面積が最小となる方向でモジュール変換を行う手順を示すフローチャート。

【図１】

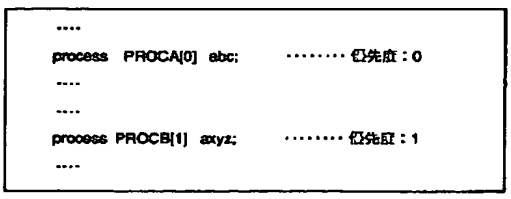


システム構成

【図２】

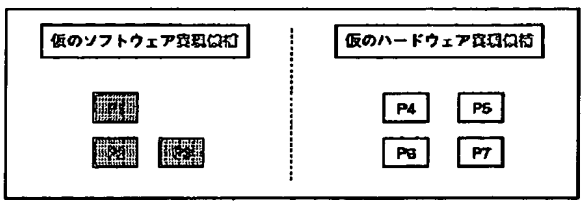
```
process     sample                    :プロセス名
input   int   iData;                  :入力データ変数
output  int   oData=0;                :出力データ変数
ctl_input     Strt;                   :制御入力信号
ctl_output    ackStrt=FALSE;          :制御出力信号
{
   int   counter=0;
   Proc  main(Strt){                  :記述の振舞ステート（StrtがTRUE
      ackStrt = TRUE;                  になると実行される）
      if((++counter) > iData)         :条件判断文
         counter = 0;                 :代入文
      oData = iData + counter;        :代入文
   }
}
```

【図３】



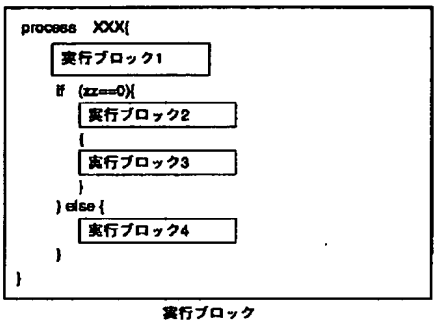
システム仕様記述におけるプロセスの優先度指定

【図４】



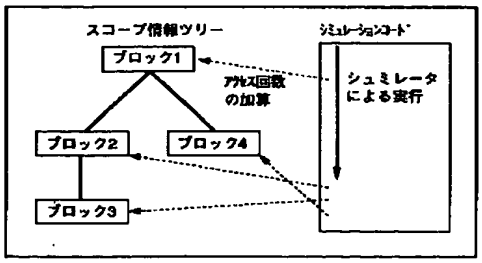
プロセスの初期分割

【図５】



実行ブロック

【図６】



スコープ情報ツリーによる変数アクセス回数の管理

【図７】

ソフトウェア実装変更候補モジュール

H1	面積 実行時間	ステップ数 実行時間	減少面積	既変	今回
H2	面積 実行時間	ステップ数 実行時間	減少面積	既変	今回
H3	面積 実行時間	ステップ数 実行時間	減少面積	既変	今回
HM	面積 実行時間	ステップ数 実行時間	減少面積	既変	今回
HM+1	面積 実行時間	ステップ数 実行時間	減少面積	既変	今回
HM+2	面積 実行時間	ステップ数 実行時間	減少面積	既変	今回

大 面積改善値

今回変更候補となった事を示すフラグ

当初計算されたデータ（初期見積りデータ）

変更した場合の換算データ

変更された事を示すフラグ

【図８】

ハードウェア実装変更候補モジュール

S1	ステップ数 実行時間	面積 実行時間	増加面積	既変	今回
S2	ステップ数 実行時間	面積 実行時間	増加面積	既変	今回
S3	ステップ数 実行時間	面積 実行時間	増加面積	既変	今回
SN	ステップ数 実行時間	面積 実行時間	増加面積	既変	今回
SN+1	ステップ数 実行時間	面積 実行時間	増加面積	既変	今回
SN+2	ステップ数 実行時間	面積 実行時間	増加面積	既変	今回

大 速度改善値

当初計算されたデータ（初期見積りデータ）

変更した場合の換算データ

変更した場合の増加面積

【図１０】

ソフトウェア実装変更候補モジュール

H1	面積 実行時間	ステップ数 実行時間	増加時間	既変	今回
H2	面積 実行時間	ステップ数 実行時間	増加時間	既変	今回
H3	面積 実行時間	ステップ数 実行時間	増加時間	既変	今回
HM	面積 実行時間	ステップ数 実行時間	増加時間	既変	今回
HM+1	面積 実行時間	ステップ数 実行時間	増加時間	既変	今回
HM+2	面積 実行時間	ステップ数 実行時間	増加時間	既変	今回

大 面積改善値

今回変更候補となった事を示すフラグ

当初計算されたデータ

変更した場合の換算データ

変更された事を示すフラグ

【図１１】

ハードウェア実装変更候補モジュール

S1	ステップ数 実行時間	面積 実行時間	減少時間	既変	今回
S2	ステップ数 実行時間	面積 実行時間	減少面積	既変	今回
S3	ステップ数 実行時間	面積 実行時間	減少面積	既変	今回
SN	ステップ数 実行時間	面積 実行時間	減少面積	既変	今回
SN+1	ステップ数 実行時間	面積 実行時間	減少面積	既変	今回
SN+2	ステップ数 実行時間	面積 実行時間	減少面積	既変	今回

大 速度改善値

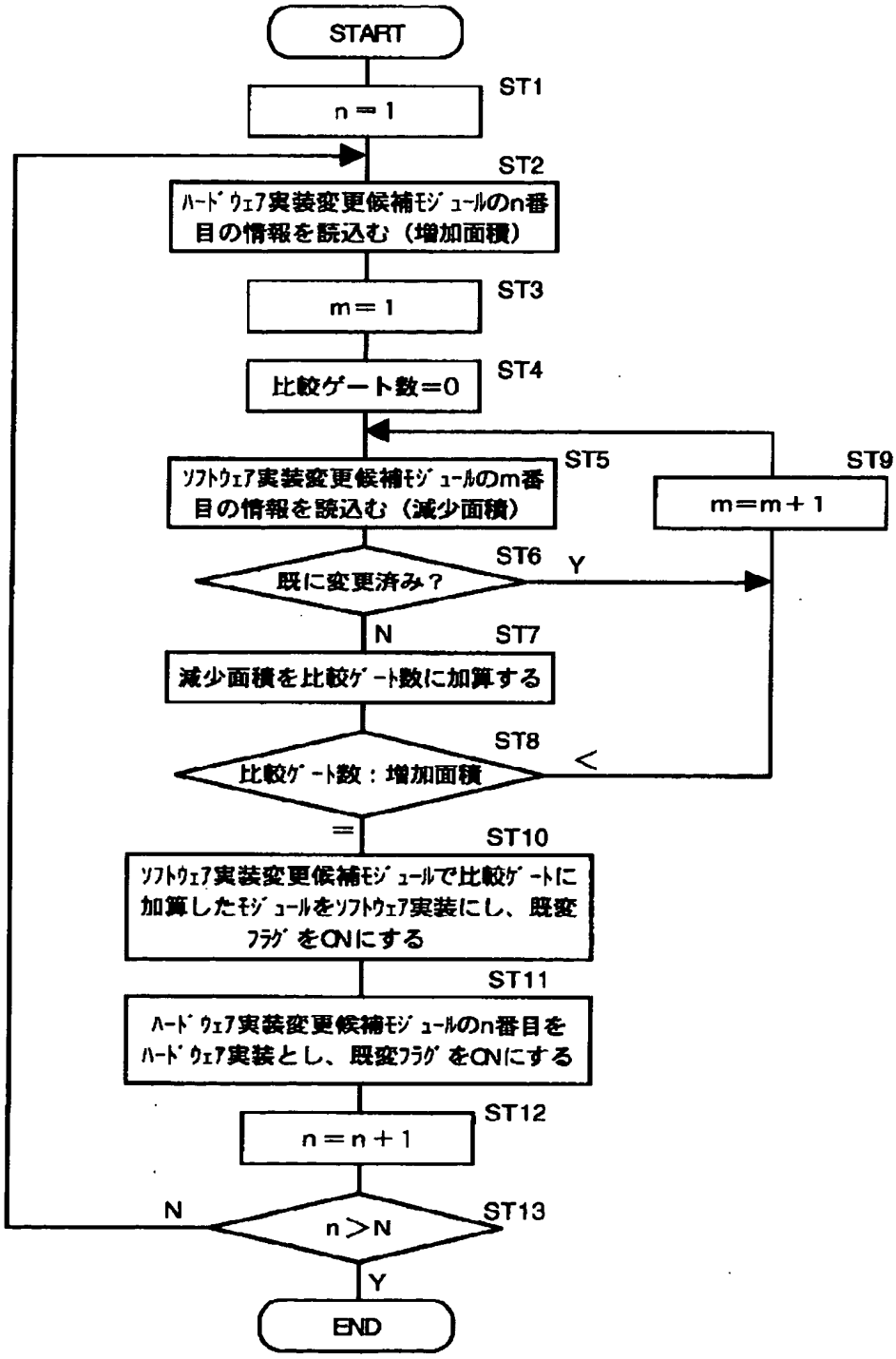
当初計算されたデータ

変更した場合の換算データ

変更した場合の増加面積

【図９】

一定の面積内で、速度が最小となる方向で変換する手順



【図１２】

面積制約がない場合

一定の速度内で、面積が最小となる方向で変換する